STUDI EMPIRIS HUBUNGAN METRIK KOHESI DENGAN KECENDERUNGAN KESALAHAN PADA APLIKASI BERORIENTASI OBJEK

Roni Yunis[1], Arwin Halim[2]

STMIK Mikroskil
Jl. Thamrin no 112, 124, 140 Medan 20212
roni@mikroskil.ac.id[1], arwin@mikroskil.ac.id[2]

Abstrak

Salah satu faktor penting dari suatu perangkat lunak adalah kualitas. Kualitas perangkat lunak yang baik ditunjukkan dengan minimalnya kesalahan-kesalahan yang ditemukan pada saat implementasi. Kesalahan yang ditemukan di awal pengembangan dapat mengurangi biaya, usaha dan waktu untuk perbaikan. Pada penelitian ini, akan diuji pengaruh antara salah satu metrik kualitas internal yaitu kohesi dan kecenderungan kesalahan pada perangkat lunak berorientasi objek. Data yang digunakan dalam penelitian berupa perangkat lunak berkode bebas yang dikembangkan dengan bahasa pemrograman Java. Pengukuran metrik kohesi dilakukan pada tahap desain dan implementasi. Hasil penelitian menunjukkan adanya korelasi positif antara metrik kualitas kohesi terutama pada kode sumber dengan kecenderungan kesalahan perangkat lunak berorientasi objek.

Kata kunci : *kecenderungan kesalahan, metrik kohesi, objek oriented*

1. Pendahuluan

Aspek kualitas merupakan salah satu aspek penting dalam pengembangan perangkat lunak. Penilaian kualitas perangkat lunak biasanya dinilai secara subjektif dan sangat dipengaruhi oleh pengetahuan perancang sistem. Salah satu cara mengukur kualitas perangkat lunak adalah dengan menghitung metrik kualitas. Metrik merupakan suatu prosedur yang memasangkan karakteristik tertentu pada entitas yang diamati menjadi sebuah nilai numerik [1]. Salah satu aspek kualitas internal yang penting pada perangkat lunak adalah nilai kohesi. Semakin tinggi nilai kohesi pada sebuah modul, maka semakin baik kualitas perangkat lunak yang dihasilkan Briand, Bunse dan Daly [2]. Metrik kualitas dapat digunakan untuk membantu perancang dalam mengukur kualitas perangkat lunak secara objektif. Salah satu penggunaan metrik kualitas adalah mendeteksi kecenderungan kesalahan seperti yang dilakukan oleh Nagappan dan Ball [3] dan Olague dkk [4]. Berdasarkan penelitian Boehm (1981) dan Jones (1996), kesalahan telah ada secara alamiah pada setiap tahap pengembangan perangkat lunak [5]. Kesalahan-kesalahan yang ditemukan pada setiap tahap pengembangan harus diperbaiki sebelum didistribusikan kepada pengguna akhir. Kesalahan yang ditemukan pada akhir pengembangan dapat meningkatkan biaya, usaha, dan waktu perbaikan. Oleh karena itu, diperlukan model yang dapat digunakan untuk mendeteksi kecenderungan kesalahan perangkat lunak.

Berbagai metrik kohesi telah diusulkan dalam menilai kualitas perangkat lunak. Metrik kohesi yang paling populer diperkenalkan oleh Chidamber dan Kemerer yang disebut metrik Lack of Cohesion Metric (LCOM). Metrik kohesi lain yang dimodifikasi dari metrik LCOM adalah LCOM2, dan LCOM3. Selain diukur dari kode program, nilai kohesi dapat dihitung berdasarkan desain perangkat lunak. J. A. Dallal telah mengusulkan metrik yang mengukur

nilai kohesivitas berdasarkan informasi yang terdapat pada tahap desain [6]. Metrik yang diusulkan telah memenuhi kriteria metrik kohesi yang baik yang telah dikembangkan oleh Briand dkk [7]. Pada penelitian ini, Penulis akan meneliti hubungan antara metrik kohesi dengan kecenderungan kesalahan perangkat lunak. Perhitungan nilai kohesi pada tahap desain dan implementasi akan dilakukan untuk setiap kelas yang terdapat pada perangkat lunak berorientasi objek. Nilai kohesi pada tahap desain diperoleh dari metrik SCC [6]. Nilai kohesi pada tahap implementasi diperoleh dari metrik LCOM dan LCOM3 (LCOM*). Hasil perhitungan nilai metrik kualitas akan diuji hubungannya dengan informasi kelas-kelas yang terdeteksi memiliki kesalahan. Pengujian hubungan antara kelas dan metrik kohesi menggunakan univariate dan multivariate regresi logistik.

2. Kajian Pustaka

2.1 Konsep Berorientasi Objek

Pendekatan berorientasi objek mulai berkembang karena adanya kesulitan untuk menghasilkan sistem yang berkualitas sesuai dengan waktu dan biaya, khususnya pada sistem skala besar yang dikembangkan banyak pengembang. Pendekatan ini memiliki perbedaan dengan pendekatan terstuktur. Pendekatan berorientasi objek menggabungkan data dan proses secara bersamaan dalam bentuk paket, sehingga sangat memudahkan pengembang dalam mengelola paket tersebut. Hal ini juga yang menjadi kelebihan dari pendekatan ini, yaitu kode program menjadi lebih mudah untuk digunakan kembali. Pendekatan berorientasi objek memperkenalkan beberapa istilah baru yaitu kelas, objek, hubungan antar kelas, dan lain-lain.

Kelas merupakan gambaran dari kumpulan objek yang memiliki spesifikasi fitur, batasan, dan semantik yang sama. Objek merupakan entitas pada program komputer yang memiliki tiga karakteristik dasar, yaitu keadaan, perilaku, dan identitas. Setiap objek pada kelas memiliki identitas yang unik, sehingga objek dapat memiliki perilaku yang berbeda-beda sesuai dengan keadaan. *Unified Modeling Language* (UML) digunakan untuk memodelkan sistem berorientasi objek seperti data, keadaan, waktu, perilaku sistem, dan lain-lain.

2.2 Metrik Kualitas

Metrik merupakan suatu prosedur yang memasangkan karakteristik tertentu pada entitas yang diamati menjadi sebuah nilai numerik [1]. Karakteristik dan entitas yang ingin diamati bersifat bebas. Oleh karena itu, manfaat metrik sangat tergantung pada apa yang akan dicapai dari hasil pengukuran yang telah dilakukan. Nilai numerik pada metrik akan memberikan pengetahuan pengamat mengenai nilai yang terlalu tinggi atau terlalu rendah, terlalu banyak atau terlalu sedikit. Dengan kata lain, metrik merupakan suatu point reference yang menunjukkan makna semantik yang berguna pada suatu nilai.

2.3 Metrik Kohesi

Nilai kohesi pada sebuah kelas menunjukkan hubungan dan interaksi antar elemen dalam kelas. Semakin tinggi nilai kohesi pada suatu kelas berarti kelas tersebut tidak dapat dipecah lagi dan memiliki sebuah fungsi yang jelas [2]. Berdasarkan penelitian dari Chen [8], nilai kohesi pada kelas akan mempengaruhi kualitas eksternal perangkat lunak seperti understandable, modifiable, dan maintainable. Class Cohesion digunakan sebagai indikator awal adanya kesalahan pada desain kelas. Metrik kohesi telah diusulkan oleh berbagai peneliti, seperti LCOM, LCOM3, dan lain-lain.

2.3.1 Lack of Cohesion (LCOM)

Misalnya sebuah kelas memiliki n operasi yaitu M_1, M_2, M_3, .., M_n. Dimana $\{I_j\}$ merupakan kumpulan variabel instan yang digunakan oleh method Mi, maka terdapat n buah himpunan, yaitu $\{I_1\}$, $\{I_2\}$, $\{I_3\}$,..$\{I_n\}$. Dimana $P = \{(I_i,I_j) \mid I_i \cap I_j = \varnothing\}$ dan $Q = \{(I_i,I_j) \mid I_i \cap I_j \neq \varnothing\}$. Jika semua n himpunan $\{I_1\}$, $\{I_2\}$, $\{I_3\}$,..$\{I_n\}$ adalah $\varnothing$, maka $P = \varnothing$. Nilai LCOM dapat dihitung dengan menggunakan Persamaan 1.

$$LCOM = \begin{cases} |P| - |Q| & \text{jika } |P| > |Q| \\ 0 & \text{jika } |P| \leq |Q| \end{cases} \quad (1)$$

2.3.2 Lack of Cohesion 3 (LCOM*)

Hitz dan Montazeri [9] mengusulkan sebuah metrik kohesi (LCOM3). LCOM3 melakukan modifikasi yang radikal terhadap LCOM. LCOM3 menyajikan hubungan antar metode pada kelas dalam bentuk graph. Metode pada kelas akan menjadi simpul. Jika dua metode menggunakan setidaknya satu atribut yang sama, maka akan ada tepi antara dua simpul yang bersesuaian. Nilai LCOM3 berada pada range 0 sampai 2. Nilai 1 sampai 2 akan memberikan peringatan rendahnya nilai kohesi pada kelas. Nilai LCOM3 dihitung berdasarkan Persamaan 2.

$$LCOM^* = (m - sum(mA)/a) / (m-1) \quad (2)$$

Dimana:

```
m : jumlah metode dalam kelas
a  : jumlah variabel dalam kelas
mA : jumlah metode yang mengakses variabel
sum (mA) : jumlah dari mA
```

2.3.3 Metrik Kohesi Desain UML

Metrik Similarity-based Class Cohesion (SCC) dikembangkan oleh Dallal [6]. Metrik SCC mengukur nilai cohesion berdasarkan interaksi antara atributatribut, metode-metode, dan atribut-metode secara langsung dan transitif. Untuk menghitung nilai metrik kohesi, informasi pada level desain akan dibentuk menjadi empat matriks interaksi, yaitu:

1. *Direct Method Invocation matrix (DMI)*
   Matrik ini menggambarkan interaksi langsung pemanggilan metode pada suatu kelas. Matrik DMI merupakan matriks persegi dengan dimensi *kxk*, dimana k merupakan jumlah metode pada kelas tersebut. Untuk setiap baris i dan kolom j pada matrik DMI akan dihitung berdasarkan Persamaan 3.

$$DMI_{ij} = \begin{cases} 1 & \text{jika metode } i \text{ memanggil metode } j \text{ secara langsung} \\ 0 & \text{jika tidak} \end{cases} \quad (3)$$

2. *Method Invocation Matrix (MI)*
   Matrik ini menggambarkan interaksi langsung dan tidak langsung terjadinya pemanggilan metode pada kelas. Informasi ini akan diperoleh dari matriks DMI. Matrik MI merupakan matriks persegi dengan dimensi kxk, dimana k merupakan jumlah metode pada kelas tersebut. Untuk setiap baris i dan kolom j pada matrik MI akan dihitung berdasarkan Persamaan 4.

$$MI_{ij} = \begin{cases} 1 & \text{jika metode } i \text{ memanggil metode } j \text{ secara langsung ataupun transitif} \\ 0 & \text{jika tidak} \end{cases} \quad (4)$$

3. *Direct Attribute Type Matrix (DAT)*
   Matrik ini menggambarkan interaksi langsung antara metode-metode, atribut-atribut, dan metode-atribut. Matrik DAT memiliki dimensi kxl dimana k merupakan jumlah metode pada sebuah kelas dan l merupakan jumlah tipe data atribut yang berbeda pada kelas. Nilai matrik dibentuk berdasarkan asumsi bahwa set himpunan tipe data yang diakses oleh metode merupakan irisan dari set himpunan dari tipe data pada parameter metode dan atribut kelas. Untuk setiap baris k dan kolom l pada matriks DAT akan dihitung berdasarkan Persamaan 5.

$$DAT_{ij} = \begin{cases} 1 & \text{jika tipe data } i \text{ merupakan tipe data dari parameter atau } \textit{return} \text{ dari metode } j \\ 0 & \text{jika tidak} \end{cases} \quad (5)$$

4. *Attribute Type Matrix (AT)*
   Matriks ini menggambarkan interaksi langsung atau transitif sesuai dengan matriks DAT dan matriks MI. Nilai 1 pada matrik AT menunjukkan adanya kesesuaian antara tipe data atribut kelas dengan tipe data pada parameter metode yang dipanggil secara langsung atau transitif. Matriks AT memiliki dimensi yang sama dengan matriks AT yang dibentuk dengan algoritma yang seperti berikut.

```
Input : Matrik DAT dan MI
Output : Matrik AT
Langkah:
For i = 1 to jlhRowDAT
  For j = 1 to jlhColumnDAT
        AT[i,j] <- DAT[i,j]
For i = 1 to jlhRowDAT
  For j = 1 to jlhColumnDAT
        If  i <> j and MI[i,j] = 1 then
              For k = 1 to jlhColumnDAT
                    AT[i,k] <- DAT[i,k] OR DAT[j,k]
```

Metrik kohesi yang diusulkan Dallal dan Briand akan dihitung berdasarkan informasi matriks yang telah dihitung sebelumnya, yaitu

1. *Method-Method through Attributes Cohesion (MMAC)*

$$MMAC(C) = \begin{cases} 0 & \text{jika } k = 0 \text{ atau } l = 0 \\ 1 & \text{jika } k = 1 \\ \frac{\sum_{i=1}^{l} x_i(x_i-1)}{lk(k-1)} & \text{jika tidak} \end{cases} \quad (6)$$

2. *Atribute-Atribute Cohesion (AAC)*

$$AAC(C) = \begin{cases} 0 & \text{jika } k = 0 \text{ atau } l = 0 \\ 1 & \text{jika } l = 1 \\ \frac{\sum_{i=1}^{k} y_i(y_i-1)}{kl(l-1)} & \text{jika tidak} \end{cases} \quad (7)$$

3. *Atribute-Method Cohesion (AMC)*

$$AMC(C) = \begin{cases} 0 & \text{jika } k = 0 \text{ atau } l = 0 \\ \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} m_{ij}}{kl} & \text{jika tidak} \end{cases} \quad (8)$$

4. *Method-Method Invocation Cohesion (MMIC)*

$$MMIC(C) = \begin{cases} 0 & \text{jika } k = 0 \\ 1 & \text{jika } k = 1 \\ \frac{\sum_{i=1}^{k}\sum_{j=1, i\neq j}^{k} m_{ij}}{k(k-1)} & \text{jika tidak} \end{cases} \quad (9)$$

5. *Similarity-based Class Cohesion (SCC)*

$$SCC(C) = \begin{cases} 0 & \text{jika } k = 0 \text{ and } l \geq 1 \\ 1 & \text{jika } k = 1 \text{ and } l = 0 \\ MMIC(C) & \text{jika } k > 1 \text{ and } l = 0 \\ \frac{k(k-1)[MMAC(C)+2MMIC(C)+l(l-1)AAC(C)+2lkAMC(C)}{3k(k-1)+l(l-1)+2lk} & \text{jika tidak} \end{cases} \quad (10)$$

2.4 Kecenderungan Kesalahan Perangkat Lunak

Berdasarkan penelitian Boehm (1981) dan Jones (1996) [5], kesalahan telah ada secara alamiah pada setiap tahap pengembangan perangkat lunak. Para pakar pengembangan perangkat lunak percaya bahwa pola distribusi kesalahan pada setiap tahap tidak akan berubah selama dua dekade. Kesalahan yang ditemukan pada tahap coding memiliki nilai paling tinggi. Hal ini dikarenakan adanya kemungkinan kesalahan saat mengimplementasikan desain. Desain yang kurang baik akan menghasilkan perangkat lunak yang kurang baik walaupun diimplementasikan dengan baik

3. Metode Penelitian

Penelitian akan dimulai dari proses mengumpulkan bahan penelitian berupa jurnal, conference, buku, dan website yang mendukung penelitian. Langkah-langkah yang dilakukan pada penelitian adalah:

1. Studi Literatur.
   Tahapan ini dimulai dengan pengumpulan bahan penelitian berupa referensi dari jurnal, *conference*, buku dan *library* yang berhubungan dengan metrik kualitas.
2. Pengembangan alat bantu ukur.
   Tahapan ini menghasilkan aplikasi alat bantu ukur nilai metrik kualitas dengan menggunakan bahasa pemrograman Java. Penulis juga memanfaatkan library ckjm yang dapat digunakan untuk membantu perhitungan nilai metrik.
3. Pengumpulan data
   Data penelitian berasal dari aplikasi berkode bebas yaitu Apache Xalan versi 2.4.0 dan 2.5.0 yang dapat diakses dari http://xml.apache.org/xalan-j/. Statistik deskriptif dari Apache Xalan 2.4 dan 2.5 ditunjukkan pada Tabel 1.

Tabel 1 Statistik Deskriptif Apache Xalan 2.4 dan Xalan 2.5

	Xalan 2.4	*Xalan 2.5*
Jumlah kelas yang diamati	628	697
Jumlah metode yang diamati	7018	7732
Total baris kode program	217120	297574
Rata-rata baris kode per kelas	345	426
Jumlah Kelas yang memiliki kesalahan	108	368

4. Pengujian hipotesa
   Tahapan ini akan menguji hubungan hasil perhitungan metrik nilai kohesi berdasarkan kode program dan desain UML dengan informasi kesalahan yang terdapat pada perangkat lunak tersebut. Hipotesa diuji dengan alat bantu SPSS untuk uji statistik non-parametric yaitu spearman rank correlation dengan tingkat kepercayaan 95%.

   Hipotesa yang akan diuji antara lain:

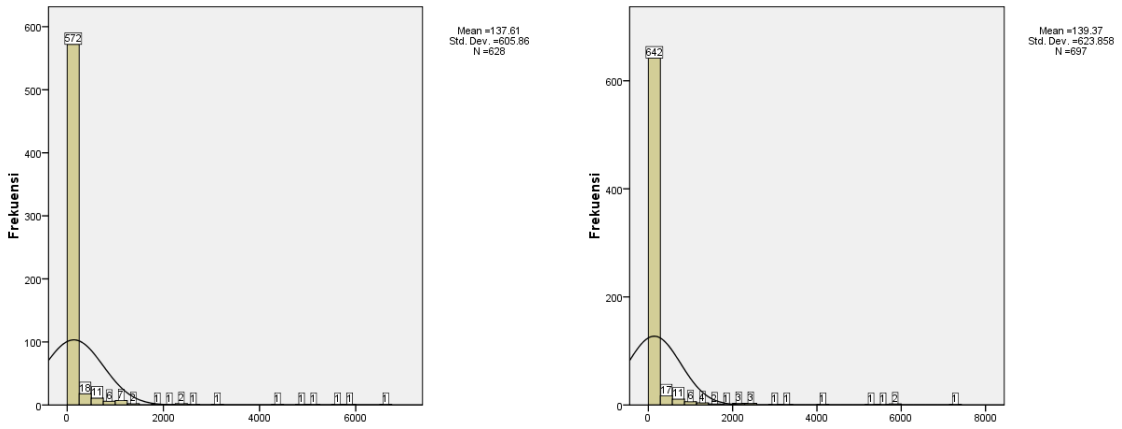
   H1 : Nilai metrik kohesi pada kode program berkorelasi positif dalam mendeteksi kecenderungan kesalahan perangkat lunak berorientasi objek.
   H2 : Nilai metrik kohesi pada desain UML berkorelasi positif dalam mendeteksi kecenderungan kesalahan perangkat lunak.

5. Penarikan kesimpulan

4. Hasil dan Pembahasan

4.1 Hasil Pengukuran Metrik LCOM

Metrik LCOM merupakan nilai yang diperoleh dari kode program. Perhitungan nilai metrik LCOM dilakukan untuk semua kelas pada Apache Xalan 2.4 dan 2.5 seperti terlihat pada Gambar 1.

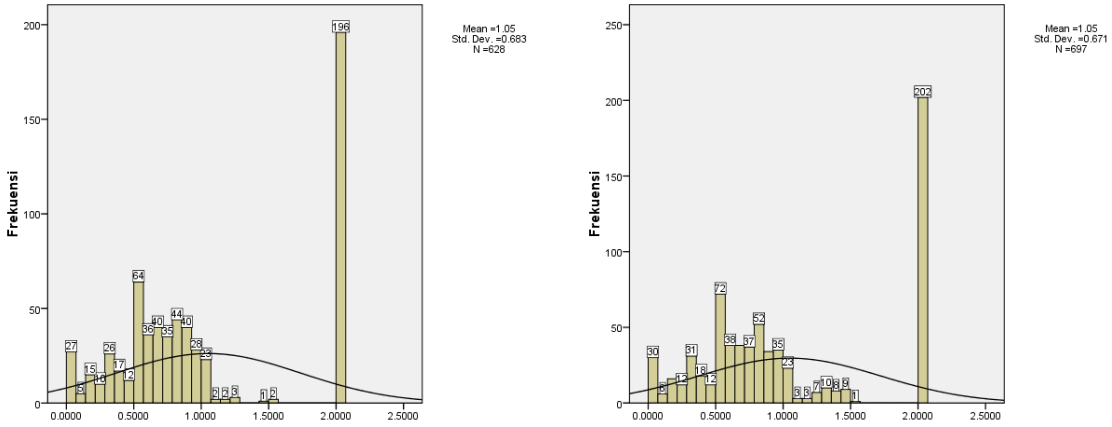


(a) (b)

Gambar 1 Distribusi Nilai LCOM pada (a) Xalan 2.4 (b) Xalan 2.5

4.2 Hasil Pengukuran Metrik LCOM*

Metrik LCOM* juga merupakan nilai yang diperoleh dari kode program. Perhitungan nilai metrik LCOM* dilakukan untuk semua kelas pada Apache Xalan 2.4 dan 2.5 seperti terlihat pada Gambar 2.

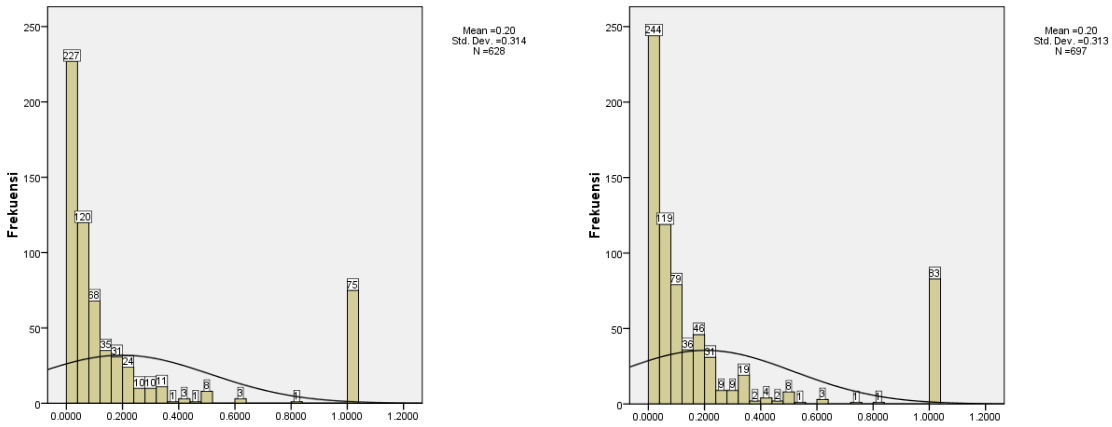


(a) (b)

Gambar 2 Distribusi Nilai LCOM* pada (a) Xalan 2.4 (b) Xalan 2.5

4.3 Hasil Pengukuran Metrik SCC

Metrik SCC merupakan nilai yang diperoleh dari diagram UML. Perhitungan nilai metrik SCC dilakukan untuk semua informasi UML pada Apache Xalan 2.4 dan 2.5 seperti terlihat pada Gambar 3.



(a) (b)

Gambar 3 Distribusi Nilai SCC pada (a) Xalan 2.4 (b) Xalan 2.5

4.4 Pembahasan

Hasil perhitungan metrik pada aplikasi Apache Xalan versi 2.4.0 dan versi 2.5.0 baik melalui kode sumber (LCOM dan LCOM*) dan desain UML (SCC), diuji korelasi dengan kecenderungan kesalahan pada kelas perangkat lunak menggunakan uji hipotesis non-parametric Spearman dengan tingkat kepercayaan 99% (1-tailed).

4.4.1 Pengujian Hipotesa Metrik Kode Program dan Kecenderungan Kesalahan

H1 : Nilai metrik kohesi pada kode program berkorelasi positif dalam mendeteksi kecenderungan kesalahan perangkat lunak berorientasi objek.

Korelasi LCOM dan kecenderungan kesalahan pada Xalan 2.4 dapat dilihat pada Tabel 2.

Tabel 2 Korelasi Nilai LCOM dan Kecenderungan Kesalahan pada Xalan 2.4

			isBug	lcom
Spearman's rho	isBug	Correlation Coefficient	1.000	.257**
		Sig. (1-tailed)	.	.000
		N	628	628
	lcom	Correlation Coefficient	.257**	1.000
		Sig. (1-tailed)	.000	.
		N	628	628

**. Correlation is significant at the 0.01 level (1-tailed).

Tabel 2 menunjukkan adanya korelasi positif antara nilai metrik kohesi LCOM dan kecenderungan kesalahan dan bersifat signifikan yang terlihat dari nilai Sig. (1-tailed) < 0.01.

Korelasi LCOM dan kecenderungan kesalahan pada Xalan 2.5 dapat dilihat pada Tabel 3.

Tabel 3 Korelasi Nilai LCOM dan Kecenderungan Kesalahan pada Xalan 2.5

			lcom	hasBug
Spearman's rho	lcom	Correlation Coefficient	1.000	.160**
		Sig. (1-tailed)	.	.000
		N	697	697
	hasBug	Correlation Coefficient	.160**	1.000
		Sig. (1-tailed)	.000	.
		N	697	697

**. Correlation is significant at the 0.01 level (1-tailed).

Tabel 3 menunjukkan adanya korelasi positif antara nilai metrik kohesi LCOM dengan kecenderungan kesalahan dan bersifat signifikan yang terlihat dari nilai Sig. (1-tailed) < 0.01.

Korelasi LCOM* dan kecenderungan kesalahan pada Xalan 2.4 dapat dilihat pada Tabel 4.

Tabel 4 Korelasi Nilai LCOM* dan Kecenderungan Kesalahan pada Xalan 2.4

			isBug	lcom*
Spearman's rho	isBug	Correlation Coefficient	1.000	-.006
		Sig. (1-tailed)	.	.444
		N	628	628
	lcom*	Correlation Coefficient	-.006	1.000
		Sig. (1-tailed)	.444	.
		N	628	628

Tabel 4 menunjukkan adanya korelasi negatif antara nilai metrik kohesi LCOM* dan kecenderungan kesalahan tetapi tidak signifikan yang terlihat dari nilai Sig. (1-tailed) > 0.01

Korelasi LCOM* dan kecenderungan kesalahan pada Xalan 2.5 dapat dilihat pada Tabel 5.

Tabel 5 Korelasi Nilai LCOM* dan Kecenderungan Kesalahan pada Xalan 2.5

			hasBug	lcom*
Spearman's rho	hasBug	Correlation Coefficient	1.000	.009
		Sig. (1-tailed)	.	.403
		N	697	697
	lcom*	Correlation Coefficient	.009	1.000
		Sig. (1-tailed)	.403	.
		N	697	697

Tabel 5 menunjukkan adanya korelasi positif antara nilai metrik kohesi LCOM* dan kecenderungan kesalahan tetapi tidak signifikan yang terlihat dari nilai Sig. (1-tailed) > 0.01.

Jadi, dapat ditarik kesimpulan nilai metrik LCOM memiliki korelasi positif dengan kecenderungan kesalahan dan bersifat signifikan, sedangkan nilai metrik LCOM* hampir tidak memiliki hubungan korelasi dengan kecenderungan kesalahan perangkat lunak. Jadi hipotesa H1 dapat diterima dengan catatan hanya metrik kohesi LCOM yang memiliki korelasi positif dan signifikan dengan kecenderungan kesalahan perangkat lunak.

4.4.2 Pengujian Hipotesa Metrik Desain dan Kecenderungan Kesalahan

H2 : Nilai metrik kohesi pada desain UML berkorelasi positif dalam mendeteksi kecenderungan kesalahan perangkat lunak

Korelasi SCC dan kecenderungan kesalahan pada Xalan 2.4 dapat dilihat pada Tabel 6.

Tabel 6 Korelasi Nilai SCC dan Kecenderungan Kesalahan pada Xalan 2.4

			isBug	SCC
Spearman's rho	isBug	Correlation Coefficient	1.000	-.051
		Sig. (1-tailed)	.	.101
		N	628	628
	SCC	Correlation Coefficient	-.051	1.000
		Sig. (1-tailed)	.101	.
		N	628	628

Tabel 6 menunjukkan adanya korelasi negatif antara nilai metrik kohesi SCC dan kecenderungan kesalahan tetapi tidak signifikan yang terlihat dari nilai Sig. (1-tailed) > 0.01

Korelasi LCOM dan kecenderungan kesalahan pada Xalan 2.5 dapat dilihat pada Tabel 7.

Tabel 7 Korelasi Nilai SCC dan Kecenderungan Kesalahan pada Xalan 2.5

			hasBug	SCC
Spearman's rho	hasBug	Correlation Coefficient	1.000	.155**
		Sig. (1-tailed)	.	.000
		N	697	697
	SCC	Correlation Coefficient	.155**	1.000
		Sig. (1-tailed)	.000	.
		N	697	697

Tabel 7 menunjukkan adanya korelasi positif antara nilai metrik kohesi SCC dan kecenderungan kesalahan dan bersifat signifikan yang terlihat dari nilai Sig. (1-tailed) < 0.01

Jadi, dapat disimpulkan bahwa nilai metrik SCC dapat digunakan sebagai salah satu prediktor kecenderungan kesalahan perangkat lunak. Hal ini terlihat dari adanya korelasi positif yang signifikan antara nilai kohesi SCC dan kecenderungan kesalahan.

5. Kesimpulan

Berdasarkan hasil penelitian, dapat ditarik kesimpulan sebagai berikut:

1. Pengukuran kualitas dapat dilakukan pada tahapan awal pengembangan perangkat lunak dengan menggunakan desain Unified Modeling Language dan kode program.
2. Nilai metrik kohesi yang diukur pada kode program mampu mendeteksi kecenderungan kesalahan perangkat lunak berorientasi objek. Nilai metrik yang diukur pada kode program khususnya metrik kohesi LCOM memiliki korelasi positif yang signifikan terhadap kecenderungan kesalahan perangkat lunak. Pada penelitian ini, metrik kualitas LCOM* yang diukur pada dua versi aplikasi open-source Apache Xalan, belum mampu menunjukkan korelasi yang signifikan.
3. Nilai metrik kualitas yang diukur menggunakan desain UML memiliki hasil yang beragam dalam mendeteksi kecenderungan kesalahan perangkat lunak. Pada Apache Xalan 2.5.0 terlihat adanya korelasi positif yang signifikan tetapi hal ini tidak didukung oleh hasil korelasi pada versi 2.4.0

Referensi

[1] Lanza, M., & Marinescu, R. 2006. Object-Oriented Metrics in Practice. Germany: Springer-Verlag Berlin Heidelberg

[2] Briand, L. C., Bunse, C., & Daly, J.W. A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. IEEE Transactions on Software Engineering, 27(6), 513–530, 2001.

[3] Nagappan, N., Ball, T., Use of Relative Code Churn Measures to Predict System Defect Density, Proceedings of the International Conference on Software Engineering, 2005

[4] H. M. Olague, L. H. Etzkorn, S. Gholston, and S. Quattlebaum. Empirical validation of three software metric suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Transactions on Software Engineering, 33(6):402-419, 2007.

[5] Galin, D. 2004. Software Quality Assurance: From Theory to Implementation. United States of America: Addison-Wesley Professional.

[6] Dallal, J. A. Mathematical Validation of Object-Oriented Class Cohesion Metrics. International Journal of Computers, 4(2), 45–52, 2010.

[7] Briand, L. C., Daly, J. W., & Wust, J. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. Proceedings Fourth International Software Metrics Symposium, 43–53, 1997.

[8] Chen, Z., Zhou Y., Xu, B., A Novel Approach to Measuring Class Cohesion Based on Dependence Analysis. Conference on Software Maintenance, pp. 377-383, 2002.

[9] Hitz, M., Montazeri, B., Measuring Coupling and Cohesion In Object-Oriented Systems. Proceedings of the International Symposium on Applied Corporate Computing, 25-34, 1997.